Journal of Biomedical Informatics 43 (2010) 752–761

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

OpenFlyData: An exemplar data web integrating gene expression data on the fruit fly *Drosophila melanogaster*

Alistair Miles [a], Jun Zhao [a], Graham Klyne [a], Helen White-Cooper [b], David Shotton [a,*]

[a] Image Bioinformatics Research Group, Department of Zoology, University of Oxford, South Parks Road, Oxford OX1 3PS, UK
[b] School of Biosciences, Cardiff University, Cardiff CF10 3AX, UK

ARTICLE INFO

Article history:
Received 12 November 2009

Keywords:
Chado
Data integration
Data web
Drosophila
Gene expression
Performance
RDF
SPARQL
Triple store
User interface

ABSTRACT

Motivation: Integrating heterogeneous data across distributed sources is a major requirement for *in silico* bioinformatics supporting translational research. For example, genome-scale data on patterns of gene expression in the fruit fly *Drosophila melanogaster* are widely used in functional genomic studies in many organisms to inform candidate gene selection and validate experimental results. However, current data integration solutions tend to be heavy weight, and require significant initial and ongoing investment of effort. Development of a common Web-based data integration infrastructure (a.k.a. data web), using Semantic Web standards, promises to alleviate these difficulties, but little is known about the feasibility, costs, risks or practical means of migrating to such an infrastructure. *Results:* We describe the development of OpenFlyData, a proof-of-concept system integrating gene expression data on *D. melanogaster,* combining Semantic Web standards with light-weight approaches to Web programming based on Web 2.0 design patterns. To support researchers designing and validating functional genomic studies, OpenFlyData includes user-facing search applications providing intuitive access to and comparison of gene expression data from FlyAtlas, the BDGP *in situ* database, and FlyTED, using data from FlyBase to expand and disambiguate gene names. OpenFlyData's services are also openly accessible, and are available for reuse by other bioinformaticians and application developers. Semi-automated methods and tools were developed to support labour- and knowledge-intensive tasks involved in deploying SPARQL services. These include methods for generating ontologies and relational-to-RDF mappings for relational databases, which we illustrate using the FlyBase Chado database schema; and methods for mapping gene identifiers between databases. The advantages of using Semantic Web standards for biomedical data integration are discussed, as are open issues. In particular, although the performance of open source SPARQL implementations is sufficient to query gene expression data directly from user-facing applications such as Web-based data fusions (a.k.a. mashups), we found open SPARQL endpoints to be vulnerable to denial-of-service-type problems, which must be mitigated to ensure reliability of services based on this standard. These results are relevant to data integration activities in translational bioinformatics. Availability: The gene expression search applications and SPARQL endpoints developed for OpenFlyData are deployed at http://openflydata.org. FlyUI, a library of JavaScript widgets providing re-usable user-interface components for *Drosophila* gene expression data, is available at http://flyui.googlecode.com. Software and ontologies to support transformation of data from FlyBase, FlyAtlas, BDGP and FlyTED to RDF are available at http://openflydata.googlecode.com. SPARQLite, an implementation of the SPARQL protocol, is available at http://sparqlite.googlecode.com. All software is provided under the GPL version 3 open source license.

1. Introduction

1.1. Drosophila gene expression data

Genetic insights from model organisms such as the fruit fly (*Drosophila* spp.) have translated into benefits for human health [1,2]. However, fundamental questions remain, and work is ongoing to characterize the function of every gene in the sequenced *Drosophila* genomes. For *Drosophila melanogaster*, genome-scale data on patterns of gene expression in time and space are publicly available. FlyAtlas,[1] for example, publishes quantitative data on tissue-specific mRNA expression levels for 26 different tissues and approximately 18,770 genes [3]. The Berkeley Drosophila Genome Project (BDGP)[2] publishes image data from *in situ* mRNA

* Corresponding author. Fax: +44 (0)1865 310447.
E-mail address: david.shotton@zoo.ox.ac.uk (D. Shotton).

[1] http://flyatlas.org/.
[2] http://fruitfly.org/cgi-bin/ex/insitu.pl.

1532-0464/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jbi.2010.04.004

hybridization experiments and DNA time-course microarray assays at different stages of embryogenesis covering 6138 genes [4,5]. FlyTED, the *Drosophila* Testis Gene Expression Database,[3] provides similar data for the adult *Drosophila* testis, currently containing 2762 mRNA *in situ* hybridization images and ancillary data revealing the patterns of gene expression of 817 genes in testes of wild type flies and seven meiotic arrest mutant strains in which spermatogenesis is defective [6]. These data are regularly consulted by research groups focused on specific aspects of organism biology, such as male infertility, to inform decisions about resource allocation and experimental design, particularly which genes or mutations to study in detail. These data are also used to validate personal experimental results, where a discrepancy, perhaps due to sample contamination, typically indicates that an assay needs to be repeated.

1.2. The need for Drosophila data integration

For any one investigation, data from several different sources, on thousands of genes, must be reviewed and compared with locally derived results. However, data repositories exhibit varying degrees of syntactic and semantic incompatibility, making it difficult to find related information scattered across several databases without searching each source individually. This inability to bring together relevant data quickly and interpret them accurately can impact a researcher's productivity and success.

Data from different sources and derived by different experimental methods reveal complimentary aspects of the underlying biology. For example, *in situ* data from BDGP show transcription of the gene *schuy* during late embryogenesis to be clearly localized in the gonad. Microarray data from FlyAtlas show *schuy* strongly transcribed in the adult testis, but not in any other tissue. In this case, the transcriptional profile established in embryogenesis appears similar to that of the adult, and the combined data strongly suggest a role for *schuy* in sperm development. These data influenced the choice of *schuy* as a candidate gene for testis *in situ* studies relating to *Drosophila* male infertility, leading to the discovery of post-meiotic transcription (previously thought not to occur in *Drosophila* spermatogenesis) and of two new classes of sub-cellularly localized transcripts dubbed "comets" and "cups" [7].

The publication in the Web, without access restriction, of these and other gene expression data, such as [8], Fly-FISH[4] [9], gene expression annotations curated by FlyBase[5] [10], and microarray studies available via NCBI GEO[6] or ArrayExpress,[7] has been highly beneficial for *Drosophila* functional genomics. Nevertheless, these data are published at separate locations and use dissimilar access methods and user interfaces, such that there is no easy way to ask questions that span the data sources.

1.3. Related work

There have been notable attempts to provide an integrated view of gene expression data for *D. melanogaster*. FlyMine[8] is an instance of the generic data warehouse platform InterMine customised for data on *Drosophila*, *Anopheles* and *Caenorhabditis* spp. [11]. FlyMine currently stores copies of embryo *in situ* data from BDGP and Fly-FISH, and of microarray data from FlyAtlas and from [8]. While it is possible to use FlyMine to view anatomy ontology annotations from BDGP side-by-side with FlyAtlas microarray data for one or more genes, this involves constructing a new query template using the generic query builder interface. The user interface is powerful, because it can be used to construct arbitrary queries over the FlyMine data model, but it can be hard to understand without an informatics background and requires time to master. Additionally, FlyMine does not provide any way to view the *in situ* images from BDGP, only their annotations. The images themselves provide more detail than the 145 anatomy ontology terms used by BDGP to annotated them [5], and are important sources of information for decision making and validation.

Approaches to biomedical data integration are reviewed by Goble and Stevens [12] and Stein [13]. These range from distribution of SQL queries over relational databases, for example the OGSA-DAI Project,[9] to data warehousing activities such as FlyMine described above. Many authors now favour Web standards, but tend to emphasize either a *service oriented* or a *data-oriented* perspective. The service-oriented perspective focuses on standards for Web service description, invocation and coordination, to enable at least semi-automated assembly and execution of computational workflows, e.g. [14–16]. The data-oriented perspective focuses on the syntax and semantics of data, to enable automated crawling, merging and reasoning over data published in the Web, e.g. [17–21]. These two perspectives on the Web as an infrastructure for sharing biomedical data are influenced by top-down work on Web standards, led by the World Wide Web Consortium (W3C). In particular, the service-oriented perspective is influenced by the Web Services Activity,[10] while the data-oriented perspective is influenced by the Semantic Web Activity. Both are also influenced by bottom-up trends in pragmatic Web developer communities, often grouped under the banner of 'Web 2.0', e.g. [22–24].

These perspectives are converging: ontologies are being used to describe the inputs, outputs and capabilities of Web services, and data from the 'deep Web' of analytical services are being exposed via Semantic Web standards [25–27]. SPARQL sits at the crux of this convergence. Although it will not satisfy all of the life science's bioinformatics requirements, SPARQL provides a standard means of making research data available to systems that need to query and analyse data from multiple sources, including both *in silico* experimental work flows and Web-based mashups. It can remove some of the "shim" software currently needed to cope with syntax and protocol differences between services, and thus provides a higher point of departure from which to tackle the challenging issues of semantic interoperability. It can also reduce the burden on data providers, because its expressiveness means that many queries can be answered 'out of the box', removing the need to design, implement or maintain complicated Web service interfaces. We thus chose to evaluate SPARQL for programmatic access to *D. melanogaster* gene expression data, and to determine whether such an infrastructure could provide the functionality, performance, reliability and scalability to underpin tools for candidate gene selection and experimental data validation in *Drosophila* functional genomics.

1.4. OpenFlyData, an exemplar data web for Drosophila gene expression data

In this paper, we describe the development of OpenFlyData [28], a proof-of-concept data web [29–31] for *D. melanogaster* gene expression data. OpenFlyData uses the Web as a data-sharing platform, employing Semantic Web standards[11] and simple HTTP protocols in the Representational State Transfer (REST) style [32], and is built from loosely coupled Open Source software components. OpenFlyData also makes use of Web 2.0 design patterns to accelerate

[3] http://www.fly-ted.org.
[4] http://fly-fish.ccbr.utoronto.ca/.
[5] http://flybase.org/.
[6] http://www.ncbi.nlm.nih.gov/geo/.
[7] http://www.ebi.ac.uk/microarray-as/ae/.
[8] http://www.flymine.org/.
[9] http://www.ogsadai.org.uk/.
[10] http://www.w3.org/2002/ws/.
[11] http://www.w3.org/2001/sw/.

prototyping of light-weight, user-facing applications that combine data from different sources 'on the fly'.

The OpenFlyData system provides two levels of functionality. The higher level is user-facing, and provides intuitive tools to search for and compare gene expression data and images, currently from three sources: FlyAtlas, the BDGP *in situ* database, and FlyTED. Data from a fourth source, FlyBase, are used to expand and disambiguate gene name queries, to link records between databases, and to provide links to relevant literature. Beneath this level are Web services, providing programmatic access to each of these data sources via the W3C standard SPARQL query language and protocol[12] [33,34]. These Web services are accessible to anyone without restriction, and will hopefully benefit other bioinformaticians and Web application developers interested in mining and integrating *D. melanogaster* gene expression data.

Our primary motivation in undertaking this work was to explore ways of minimizing the time it takes a researcher with no informatics background to find and compare gene expression data from different databases on a large number of genes. However, since public funding for primary databases is stretched [35,36], a parallel motivation was to explore architectures for data sharing, publication and integration that are robust and scalable, yet can be developed, sustained and adapted to emerging requirements at relatively low cost and effort; and that could be developed in small increments by a distributed and loosely coordinated community, thus spreading the costs.

2. System and methods

2.1. OpenFlyData – system overview

We developed a set of OpenFlyData Web applications providing search and comparison of gene expression data from FlyAtlas, BDGP and FlyTED, using data from FlyBase to expand and disambiguate gene name queries, link records between databases, and provide links to relevant literature. These user-facing applications query data derived from the four sources via Web service endpoints, one for each data source, implementing the W3C standard SPARQL query language and protocol and thus supporting arbitrarily complex queries over the underlying data. (A Web service endpoint implementing the SPARQL protocol is known as a 'SPARQL endpoint'.[13]) These endpoints are openly accessible for third-party use, and are deployed, together with the user-facing applications, at http://openflydata.org.

2.2. User-facing cross-database search applications

Three main user-facing Web applications are currently deployed at openflydata.org. Each provides a different capability: search by gene, search by gene batch, and search by tissue expression profile.

The first application (see Fig. 1)[14] provides the ability to search for a single gene of interest, and then to retrieve and display in a single Web browser window expression data for that gene, including tissue-specific mRNA levels from FlyAtlas, embryo *in situ* hybridization images and ontology annotations from BDGP, and testis *in situ* hybridization images from FlyTED. Tissue-specific mRNA expression level data are displayed for each of the one or more Affymetrix *Drosophila* 2.0 probes corresponding to the selected gene. These data are presented following the table layout used on the FlyAtlas Web site, but with added row background colours used to differentiate tissues where expression is either up, down, or unchanged with respect to the whole fly, and with bold font used to indicate tissues where levels are either >2 or <0.5. Links are provided to probe and gene summaries at flyatlas.org. Images from BDGP of *in situ* hybridization in embryos are displayed as thumbnails, together with anatomy ontology annotations, and are grouped by developmental stage, with links to full size images and the gene report at fruitfly.org. Images from FlyTED of *in situ* hybridization in testes are also displayed as thumbnails, with captions indicating the genotype of the sample (e.g. 'wt' is wild type, 'tomb' is the tombola meiotic arrest mutant strain). Links are provided to full size images and reports at fly-ted.org. Also retrieved are the most recent publications relevant to the selected gene, provided by FlyBase.

The "Gene Finder" widget of the application takes the input query string to search for a single known gene, using gene identifier and synonym data from FlyBase. If the query matches a single gene, a further search is immediately triggered for gene expression data and literature references for that gene. If the query matches more than one gene, the possible matches are displayed as a disambiguation list, and the user must select one of the alternatives in order to proceed.

This application is intended to cut down the time required to retrieve and compare data from these four sources, time that would otherwise be spent interacting with four different Web sites and manually resolving differences in query interfaces and gene nomenclature.

It is often useful to compare expression data for more than one gene, to look for commonalities or patterns. The second application[15] provides the ability to search for a batch of genes, and then to retrieve and display expression data for all matching genes in a compact form. For this application, the user first enters a set of gene names, identifiers or synonyms, into a text area. When this query is submitted, the application attempts to find known genes, again using gene identifier and synonym data from FlyBase. A summary is then shown indicating which of the query terms could be resolved unambiguously to a single gene, which were ambiguous, and which could not be matched to any known gene. For the set of genes which were found unambiguously, a further search is immediately triggered for gene expression data from the three other sources. Data from FlyAtlas are displayed as a compact table, with one cell per probe/tissue combination. Each cell contains the mean mRNA level and the standard error on the mean. The background colour of each cell indicates whether the level is up, down, or unchanged with respect to the whole fly. This table thus provides a crude 'heat map' visualization of the data. Embryo *in situ* images are displayed as smaller thumbnails, with anatomy ontology annotations, again grouped by stage, but arranged horizontally rather than vertically (i.e. one row per gene), facilitating comparison of data for multiple genes. Testis *in situ* images are similarly arranged.

When designing a new biological investigation, it is often useful to search for candidate genes based on their expression characteristics. The third application[16] provides the ability to search for genes matching a given tissue-specific mRNA expression profile (see Fig. 2), and then to retrieve further expression data for each gene found. As shown in Fig. 2, drop-down boxes are used to define an expression profile: for each tissue, the user indicates whether expression should be up, down, or unchanged with respect to the whole fly, or whether it could be any of these (e.g. the user could ask for genes where expression is up in testes and down in all other tissues). When this query is submitted, the application first retrieves the set of Affymetrix *Drosophila* 2.0 probes matching the query profile, and then uses links between probes and genes to retrieve a list of all corresponding

[12] http://www.w3.org/TR/rdf-sparql-query/.
[13] http://semanticweb.org/wiki/SPARQL_endpoint.
[14] http://openflydata.org/search/gene-expression.
[15] http://openflydata.org/search/gene-batch-expression.
[16] http://openflydata.org/search/byexpression-profile.

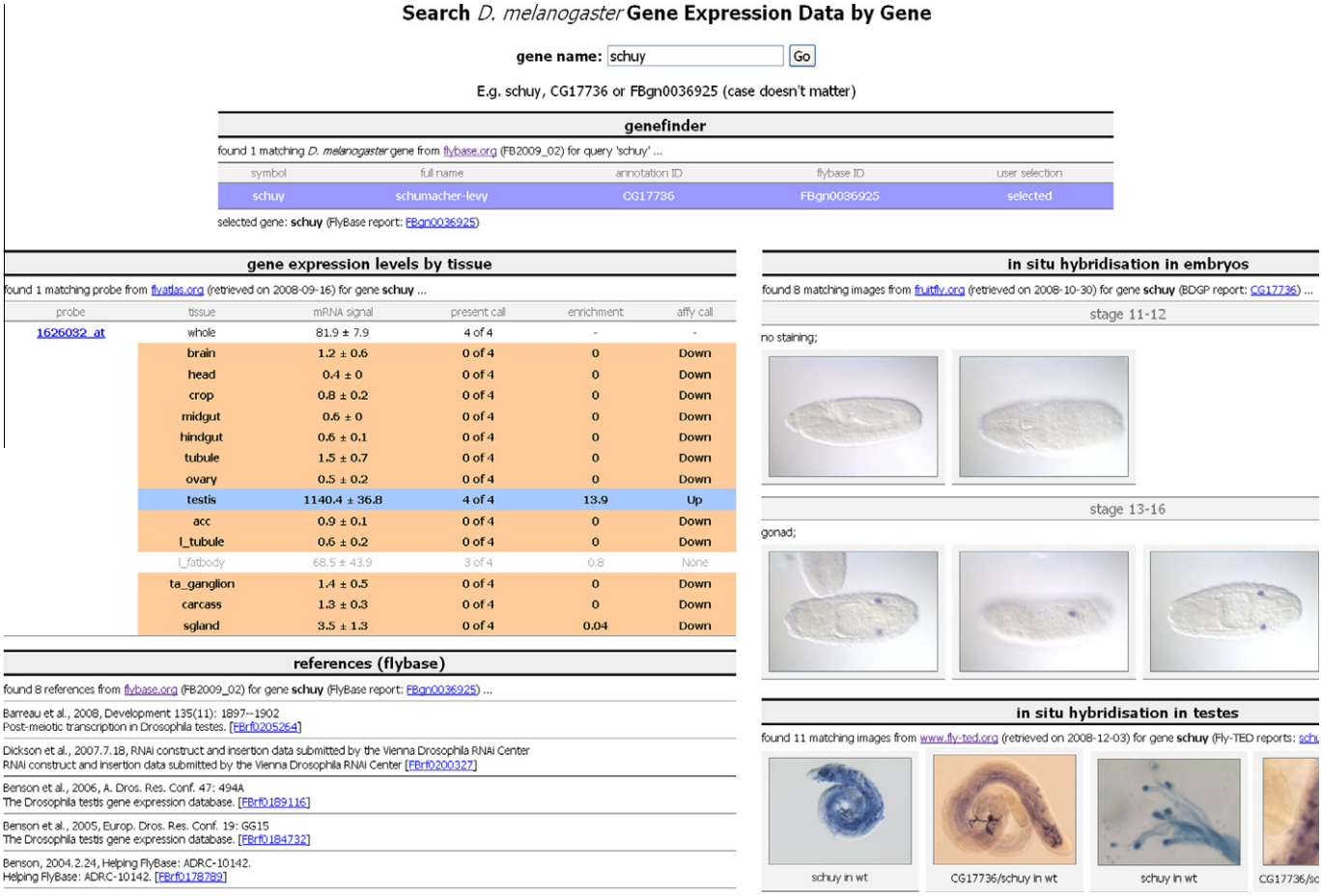

Fig. 1. A screenshot of the first OpenFlyData application that searches for gene expression data by gene name. This screenshot shows the OpenFlyData search-for-gene-expression-by-gene-name application user interface, after having conducted a search on the gene name '*schuy*'. In the upper part of the figure, below the search box, the Gene Finder widget displays a single unambiguous result from the gene name disambiguation query made against FlyBase for the name '*schuy*', which corresponds to gene with the FlyBase ID FBgn0036925, also known as CG17736 and *schumacher-levy*. Below this, data has automatically been returned to the user's browser by four other OpenFlyData widgets, clockwise from top left: (a) From FlyAtlas: quantitative data showing gene expression levels of this gene in different *Drosophila melanogaster* body tissues. Data for testis are shown highlighted in blue, because the level is elevated relative to the whole fly, while those for almost all the other tissues are shown on a peach orange background, indicating that these levels are reduced relative to the whole fly. (b) From BDGP: *in situ* hybridization images of *D. melanogaster* embryos, showing late expression in stage 13–16 embryos restricted to the gonads. (c) From FlyTED: *in situ* hybridization images of adult testes of wild type *D. melanogaster*. Other images, not shown, show the expression of the gene in two meiotic arrest mutants, *topi* and *comr*. (d) From FlyBase: bibliographic references to journal articles of relevance to the gene *schuy*.

genes from FlyBase. The user can additionally select any of the genes found, and retrieve the full tissue-specific mRNA data, and both embryo and testis *in situ* data, for that gene.

Additional screen shots from these applications are given in Supplementary Information File S1.

2.3. FlyUI – re-usable user-interface components

Each of the Web applications described above is a dynamic HTML/JavaScript rich client application that runs in a Web browser (tested in Firefox 3, IE 6 and 7, and Safari 3.2). Each application was constructed by composing two or more "widgets" drawn from FlyUI, a library of custom JavaScript widgets developed specifically for interaction with and visualization of *Drosophila* functional genomic data. Each widget fetches data on-demand via asynchronous HTTP GET or POST requests to the SPARQL endpoints described below. We developed a number of design patterns for FlyUI to make each widget more testable, and to make the process of composing widgets into one or more applications as simple as possible. Briefly, the internal architecture of each widget follows a strict model-view-controller structure, a design model usually seen only in server-side software deployments. Events triggered by user interaction with the widget's portion of the browser's document object model (DOM) are mapped to method calls on the *controller* by a *user-event handler*. The *controller* typically makes an asynchronous request for data, then updates the widget's *model* on receipt of the response. A *renderer* listens for changes to the *model* and updates the widget's DOM. Renderers for each widget are pluggable, so a new renderer may easily be written to visualize data in a different way (e.g. as a chart rather than a table). The details of constructing a SPARQL request and parsing the response are also abstracted via a pluggable *service* component, so FlyUI widgets could be adapted to use any Web service interface (such as that provided by FlyMine). Each widget exposes a direct manipulation interface, so that an application can initiate specific widget actions, and a publish-subscribe interface, so that applications can register for and be notified of events in a widget's life cycle (such as receipt of data) and thus coordinate behaviour between a set of widgets.

2.4. Web services – SPARQL endpoints

A SPARQL endpoint has been deployed for each of the three sources of experimental gene expression data (FlyAtlas, BDGP *in situ* database and FlyTED), and for the majority of data held by

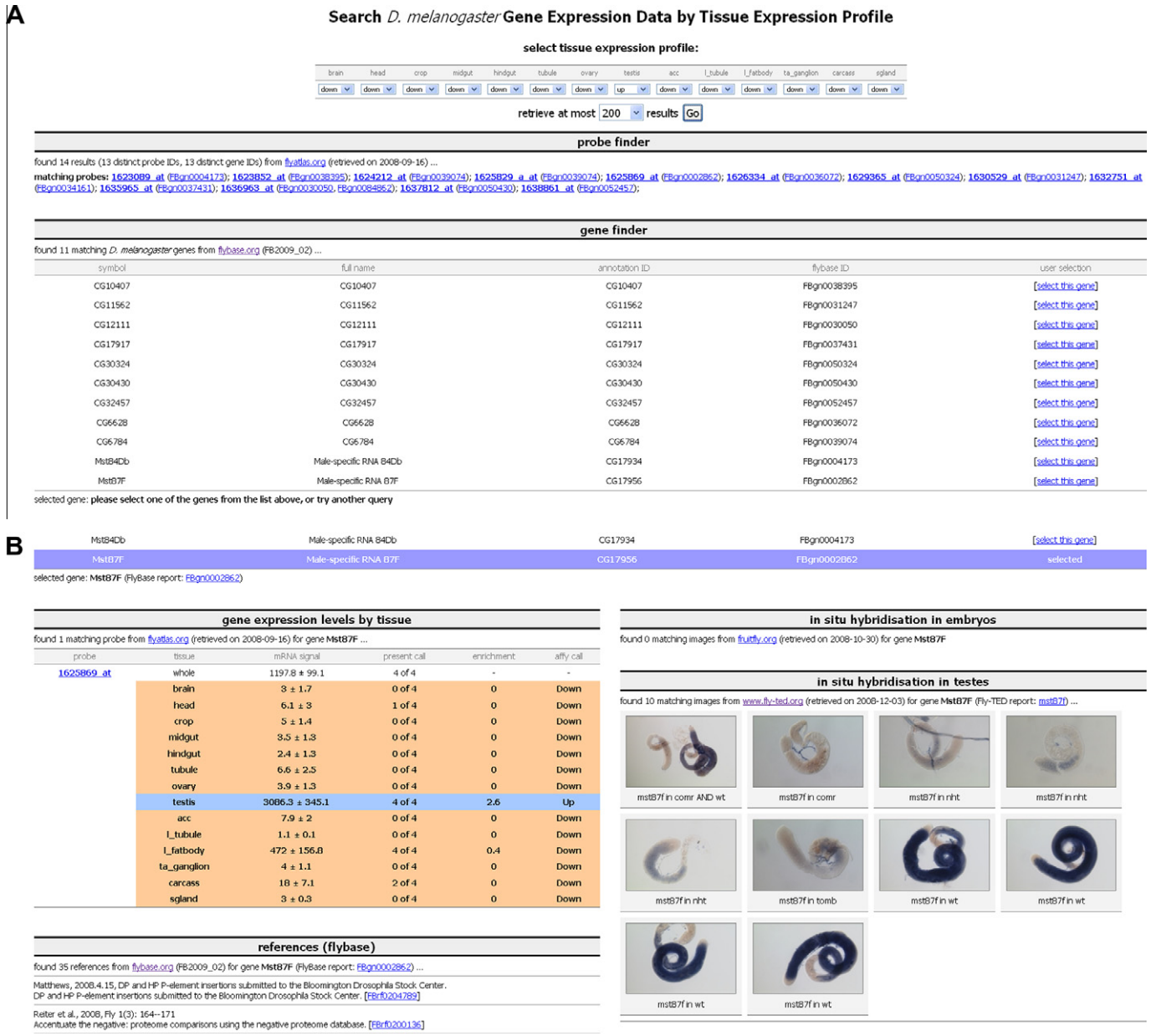

Fig. 2. The third OpenFlyData application that searches for gene expression data by tissue expression profile. (A) The gene results returned after a search to find genes whose expression is elevated in testis and reduced in other tissues, relative to the whole fly. The application's Probe Finder widget has retrieved and displayed a set of thirteen Affymetrix *Drosophila* 2.0 probes matching the query profile. Two probes, 1624212_at and 1625829_a_at, both identify gene FBgn0039074, while probe 1636963_at identifies two genes, FBgn0030050 in *Drosophila melanogaster* and FBgn0084862 in *Drosophila simulans*. Clicking on one of the returned probe IDs would take the user to full information in FlyAtlas on that probe, including its sequence, while clicking on the gene ID would take the user to full information in FlyBase on that gene. The OpenFlyData application's Gene Finder widget (the same as employed in the other applications and shown in Fig. 1) then automatically submits the gene names associated with the probes to FlyBase for disambiguation, and returns a list of the 13 *Drosophila melanogaster* gene names identifies by the probes, together with their synonyms and identifiers. At this stage the application pauses for the user to choose a gene of interest. (B) A screen shot of the lower half of the screen resulting from the user choosing to request gene expression information on the gene *Mst87F* (FBgn0002862), the last in the list. Clicking on the gene name triggers a search for quantitative gene expression data from FlyAtlas, *in situ* gene expression images from BDGP and FlyTED, and bibliographic references from FlyBase, exactly as when undertaking a single gene search (Application One, Fig. 1). The results show images returned from FlyTED of the expression of this gene in the adult testis of wild type and three meiotic arrest mutants (*comr*, *nht* and *tomb*) of *D. melanogaster*, but there are no BDGP images of the expression of this gene in *Drosophila* embryos.

FlyBase (which includes curated gene identifier and synonym data). The size of the Resource Description Framework (RDF) datasets behind these endpoints ranges from ~30,000 triples (FlyTED) up to ~175 million triples (FlyBase). Each endpoint implements the Representational State Transfer (REST) binding of the SPARQL protocol, but with some limitations (see Section 3.2 below). See Supplementary Information File S2 for examples of SPARQL queries used during a typical user session with one of the applications described above. To provide quality assurances for these SPARQL endpoints, we first defined our expectations for the structure and content of each dataset via a set of test cases. Each test case consisted of a set of SPARQL ASK queries and an expected outcome (either true or false). Each ASK query defines a pattern which we either do or do not expect to find in the dataset. Typically, such tests are probing for the presence, absence or specific cardinality of a given property on a given type of resource. Example test queries are given in Supplementary Information File S3.

The overall OpenFlyData systems architecture is shown in Fig. 3.

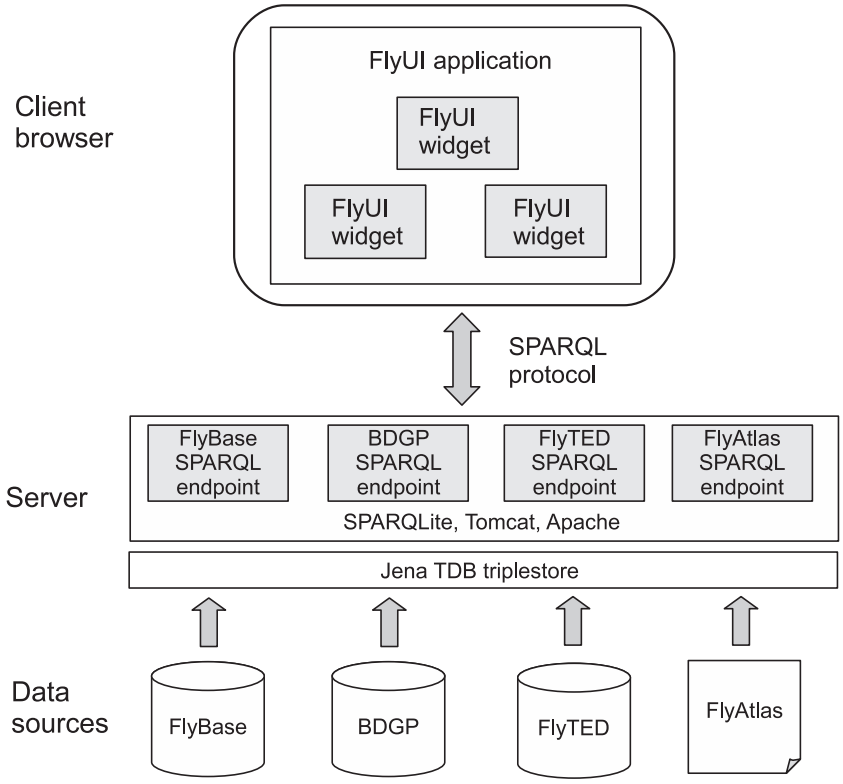

Fig. 3. The overall systems architecture of OpenFlyData. Data from each of the four databases shown at the bottom of the figure are transformed into RDF, stored in a Jena TDB triple store, and exposed by the OpenFlyData server as a SPARQL endpoint. Communication between the user and the OpenFlyData server takes the form of SPARQL queries sent to the server using the SPARQL protocol and RDF results transmitted back to the client browser via standard HTTP web exchanges. The rich OpenFlyData client employs JavaScript and FlyUI widgets to position particular types of data in defined ways in specific regions of the browser window. A version of this figure was first published in [28], and it is used with permission.

2.5. Data transformation

Deploying a SPARQL endpoint for a dataset requires a mapping from the data in its source format to the Resource Description Framework (RDF) [37]. RDF is the graph based data formalism underpinning the SPARQL query language, analogous to the way that the relational model underpins SQL. RDF is designed for linking and merging distributed data, and thus is well-suited to our situation where data from four sources are being used in combination.

2.6. Creating SPARQL endpoints

Once a mapping to RDF is defined, two alternative strategies are available, which we call *RDF caching* and *SPARQL virtualization*. The RDF caching strategy uses the mapping to obtain a dump of the data in a concrete RDF syntax such as N-TRIPLES.[17] This dump is then loaded into an off-the-shelf RDF storage system (a.k.a triple store, e.g. Jena TDB[18]) and connected with a SPARQL server (e.g. Jena Joseki[19] or SPARQLite,[20] our own SPARQL protocol implementation developed to address quality-of-service issues discussed in Sections 3.2 and 3.3 below), where it can be directly interrogated by an incoming SPARQL query. When programmatic access to the source database is available, the alternative strategy, SPARQL virtualization, can be employed. Here, software tools such as D2R Server[21] may be employed to rewrite SPARQL queries to queries against the source data in its native format. For example, where the data are available as a relational database, SPARQL queries are rewritten to SQL, and the relational database engine is used to evaluate the query. Note that both strategies permit Semantic Web applications to work over non-RDF data sources. In OpenFlyData, RDF caching was used for all our four databases.

FlyBase provides direct SQL access to a public instance of its relational database. The BDGP *in situ* database provides a downloadable SQL dump. For each of these, a mapping to RDF was defined using the D2RQ language.[22] For BDGP we adapted previous work by Mungall.[23] These mappings were then used to generate RDF dumps in N-TRIPLES syntax, via the D2R Server[24] dump-rdf utility. We also evaluated SPARQL virtualization with D2R Server for both FlyBase and BDGP, but encountered scalability limitations, and thus opted for RDF caching.

FlyBase is an implementation of the Chado schema, a generic relational schema for model organism databases developed by the Generic Model Organism Database project (GMOD) [38]. Chado is a relatively complex structure, and defining a mapping to RDF via manual editing of D2RQ mapping files is both time-consuming and error-prone. To alleviate these issues, we adopted a semi-automated strategy, whereby annotations on the Chado SQL schema definition files were used to generate both the D2RQ mapping files and a suitable Web Ontology Language (OWL)[25] ontology (see below).

[17] http://www.w3.org/TR/rdf-testcases/#ntriples.
[18] http://openjena.org/TDB/.
[19] http://www.joseki.org/.
[20] http://sparqlite.googlecode.com/.
[21] http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/.
[22] http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
[23] http://www.bioontology.org/wiki/index.php/OBD:SPARQL-InSitu.
[24] http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/.
[25] http://www.w3.org/TR/owl-features/.

FlyAtlas provides a tab-delimited download of its microarray data in a bespoke layout, which was transformed to an RDF dump via a Python conversion script. Linking probe identifiers in the raw microarray data to *D. melanogaster* gene identifiers requires probe annotation tables from Affymetrix, also available as a tab-delimited download; these too were transformed using a Python script, and the resulting data merged with the FlyAtlas RDF dump.

FlyTED provides access to image metadata via the OAI-PMH protocol. A custom Java program had been previously written to extract the metadata via this protocol and generate an RDF dump [39].

2.7. Ontology design

A relational schema defines the table and column structure of a relational database. By rough analogy, OWL ontologies define classes and properties for use in structuring RDF data. Transforming data to RDF thus involves choosing and/or developing the required OWL ontologies. We chose to adopt a conservative, incremental approach, whereby each data source was first considered in isolation. For each data source, one or more new OWL ontologies were developed to reflect the structure of the data in its source format, that could therefore provide a target for a straightforward transformation of data to RDF. The rationale for this approach is that it takes time to comprehend the precise intended semantics of data from different providers, and early attempts to completely align these data sources can thus be prone to misinterpretation. For the same reason, we have not yet attempted to map the data onto existing biomedical ontologies such as the Sequence Ontology (SO),[26] the Gene Ontology (GO),[27] or the Drosophila Anatomy Ontology,[28] except where those ontologies are explicitly cited in the source data. No combination of existing ontologies was found that provides complete coverage of the content of our selected data sources.

OWL ontologies were developed for each of FlyAtlas, FlyTED and the BDGP *in situ* database. For FlyBase, a layered collection of OWL ontologies was developed, centred on a projection of the Chado relational schema to OWL (Chado-in-OWL[29]). For the relational data sources (FlyBase and BDGP) we developed a semi-automated methodology for deriving an OWL ontology with a minimum of interpretation, yet which is relatively intuitive and convenient to query. Briefly, each table in the relational schema is first characterized as either an *entity* or an *association*. Each *association* table is further characterized as either a *binary association* or an *n-ary association*. A class is generated in the output ontology for each *entity* or *n-ary association* table. Each column in an *entity* or *n-ary association* table is characterized either as a *data attribute*, which is projected to the output ontology as a datatype property, as a *link attribute*, which is projected to the output ontology as an object property, or as a *metadata/meta-modelling attribute*, which is not projected to the output ontology. For each *binary association* table, one column is characterized as the *source* and another as the *target*, and a property is generated in the ontology with domain and range corresponding to constraints on the *source* and *target*. A worked example is provided for the FlyBase Chado schema in Supplementary Information File S4.

2.8. URI design

RDF, OWL and SPARQL require that uniform resource identifiers (URIs) be used to refer to entities (individuals), types of entities (classes) and types of relationship (properties). Choices therefore have to be made regarding which URIs to use, or if no appropriate URIs have already been coined, how new URIs will be constructed. Ideally, data providers would form consortia to establish shared URIs for biomedical entities. There are initiatives in this direction,[30] but currently only a minority of data providers use URIs, not including FlyBase, BDGP or FlyAtlas. For the purposes of this project, we established a single URI namespace under the openflydata.org domain, and constructed URIs for entities using the general pattern

`<http://openflydata.org/id/{provider}/{type}/{id}>`

where {provider} is a short name for the data provider providing the primary description of the entity (e.g. 'flybase'), {type} is a short name for the scope within which identifiers from that provider are unique (typically corresponding to a basic entity type, e.g. 'feature'), and {id} is an identifier derived from the source data, e.g. a database primary key value or some combination of unique values from a given table. For example, the URI `<http://openflydata.org7/id/flybase/stock/FBst0000009>` denotes the Bloomington *Drosophila* Stock Centre stock number 9, where 'FBst0000009' is the value of the unique column 'uniquename' in the 'stock' table of the FlyBase database. Other cases are less straightforward, e.g. the 'feature' table in the FlyBase database has a unique constraint on a combination of three columns: the 'organism' to which the sequence feature belongs, a 'type_id' for the Sequence Ontology feature type, and a 'uniquename'. A combination of these values was used to construct feature URIs, e.g. `<http://openflydata.org/id/flybase/feature/Drosophila_melanogaster/SO_ 0000704/FBgn0036925>` denotes the *D. melanogaster* gene identified by FlyBase as 'FBgn0036925'. We intend to migrate to URIs owned and maintained by suitable authorities as they become available.

3. Implementation

3.1. Syntactic and semantic interoperability – issues in linking data

When integrating data from separate resources, three problems have to be overcome. Because data are stored in incompatible formats within different database management systems, there are syntactic differences between data sources, that we overcome by core data to RDF, accessible using SPARQL. There are also semantic differences between data sources (e.g. one person's "author" is another person's "creator"), which can be solved by mapping to a common data schema or ontology. Finally there is the co-reference problem, when the same entity is known by different names in different databases (synonyms), or different entities are ambiguously described by the same name (homonyms). For *Drosophila* genetics, the ambiguity of gene names present in the data integrated within OpenFlyData presented particular problems, the resolution of which is described separately in Supplementary Information File S5, since, while the problem is a generic one, the particulars of the solutions we used, and how we arrived at them, are not central to understanding of the data web concept described in this paper.

3.2. SPARQL performance, scalability and quality-of-service

Query performance in OpenFlyData was sufficiently rapid to use SPARQL endpoints directly in browser-based data fusions (mashups) for almost all queries required in our gene expression applications. All SPARQL requests made by the single gene search application complete in ~100–500 ms (round trip time, depending on client location and connection speed), making the application as

[26] http://www.sequenceontology.org/.
[27] http://www.geneontology.org/.
[28] http://www.obofoundry.org/cgi-bin/detail.cgi?id=fly_anatomy.
[29] http://purl.org/net/chado/schema/.
[30] http://sharedname.org/.

a whole very responsive. The SPARQL queries required for the gene batch search application are more demanding, and can take up to 10 s, which is around the upper limit for reasonable usability. SPARQL queries used by the expression profile search application can be slower than 10 s, and thus optimization strategies may be needed to achieve reasonable usability. Data comparing performance of the openflydata.org FlyBase SPARQL endpoint with the FlyBase relational database are given in Supplementary Information File S6.

There is a fundamental trade-off for any service implementing the SPARQL protocol and query language. An advantage of SPARQL over a bespoke Web service interface is its inherent expressiveness and flexibility. However, it is possible to submit queries that are 'expensive', requiring a significant amount of some computational resource (e.g. main memory). If resource usage is not managed by the service implementation, denial-of-service-type problems can arise if clients submit expensive queries (accidentally or intentionally), causing poor performance or service non-availability.

3.3. SPARQLite

To explore strategies for mitigating SPARQL quality-of-service issues mentioned above, we developed SPARQLite, an implementation of the SPARQL Protocol for RDF.[31] SPARQLite has the following design goals:

- *Scalability:* There should be no bottlenecks or other impediments that prevent use of the service for many concurrent requests with arbitrary SPARQL queries over billion-triple datasets.
- *Quality-of-service:* It should be possible to manage load and resource usage, so that predictable levels of performance can be maintained over long periods of time.
- *Reliability:* It should not be vulnerable to denial-of-service-type problems or anything else that might compromise quality-of-service.
- *Dynamic configuration:* It should be possible to create, configure and destroy endpoints dynamically, without needing to restart the context or the container.

SPARQLite is a Java web application, intended for deployment within a servlet container such as Tomcat, and currently uses Jena TDB as the underlying RDF storage and query system. SPARQLite version 0.4 includes support for TDB named graph datasets, for the ARQ query language, and for LARQ (Lucene + ARQ) integration.

Together with TDB, use of SPARQLite, which also works in a streaming fashion, removes some obvious memory bottleneck vulnerabilities. SPARQLite also implements some simple strategies for limiting resource consumption, including limiting the query language features supported by a given endpoint (e.g. disallowing FILTER clauses and/or variable predicates) and by imposing a LIMIT ceiling. These all contribute to providing acceptable performance for OpenFlyData. A better strategy might be to provide resource-usage-based limitations on query evaluation, e.g. a time-out, beyond which query evaluation is cut off, coupled with limits on the maximum number of active queries for any given endpoint, but these features have yet to be implemented.

4. Discussion

4.1. Future work

OpenFlyData is a proof-of-concept system, exploring issues in the development of usable, scalable and sustainable biomedical data integration tools that might be of considerable usefulness in translational bioinformatics. Four developments would need to take place for this system to be of genuine usefulness to *Drosophila* researchers, and to become sustainable.

First, the diversity and density of data and services that are compliant with SPARQL and RESTful protocols would need to be expanded. We intend to include more *Drosophila* data into OpenFlyData, specifically from ArrayExpress and recently entered data from FlyAtlas, and to build additional applications over the data, allowing *Drosophila* researchers to use numerical gene expression information in more flexible ways to drive their exploration of genes of interest. Another way to achieve this expansion is by community take-up, spreading the responsibility and effort for ongoing maintenance and further development.

Second, OpenFlyData requires enhanced performance to cope with increased data and usage. For at least some queries, speed of SPARQL query evaluation is similar to, and may be better than, relational database engines executing comparable SQL queries (see Supplementary Information File S6, and also [40]). However, the existing open SPARQL endpoints are vulnerable to denial-of-service-type problems, which can be exposed unintentionally by bioinformatics applications that pose relatively complex queries. Current open source SPARQL implementations provide limited support for mitigating such problems.

Third, to accommodate the increasing density and diversity of data being integrated, an expanded OpenFlyData would require further work on design and implementation of powerful yet intuitive user interfaces for querying and visualising the data. This could include faceted browsers such as Exhibit,[32] that enable users to dynamically customize those data they wish to see brought together in any one view, by selection along a number of semantic axes.

Finally, to move from proof-of-concept to production services, and to catalyse the formation of a community of participation, OpenFlyData would require some funding support. This support could come either directly from a funding agency or indirectly in terms of contributions of effort and expertise by the participating *Drosophila* data providers. The need to fund data management infrastructure has of late been increasingly acknowledged by funding agencies and supported by journals [18,19,41], but such funding is unlikely to be forthcoming without clear and emphatic community support.

4.2. Transferring the data web approach to other domains

4.2.1. Use of Semantic Web technologies

For take-up of Semantic Web standards to be worthwhile in any domain, there must be short-term benefits. Our experience with *Drosophila* gene expression data suggests a short-term value proposition for SPARQL as a programmatic interface to any biomedical data, since off-the-shelf open source SPARQL implementations are available, removing the need for server-side programming to implement Web services for access to biomedical data. SPARQL services can also be integrated with other Web service frameworks and are convenient to invoke from within browser-based 'mashup' applications.

4.2.2. Mapping data to RDF

Mapping data to RDF requires expertise and effort, for which there is at present little or no tool support. Also, there are few case studies or publications describing principles or methodologies to guide the design of such transformations, although see e.g. [42]. This conversion step thus remains a significant bottleneck to the

[31] http://www.w3.org/TR/rdf-sparql-protocol/.

[32] http://code.google.com/p/simile-widgets/wiki/Exhibit.

availability of biomedical data sources as RDF, although projects like Bio2RDF [21] are ameliorating this. Inspired by model-driven software engineering (MDE) [43], we have developed some primitive, semi-automated strategies for generating such mappings, together with their supporting tests and ontologies. Further work is needed to provide robust, generic support for these tasks within existing MDE tools.

Based on our experience of transforming four different *Drosophila* databases into RDF, we propose the following general methodology for transforming existing data to RDF:

1. Decide on the transformation strategy, choosing between RDF caching and SPARQL virtualization.
2. Construct schemas or ontologies closely based on the source data schemas, imposing as little interpretation as possible and reusing existing ontologies wherever possible.
3. Work towards shared URIs for biomedical entities at an early stage. Design a URI naming convention, following W3C guidance[33] and best practice, employing namespaces that are under your control.
4. Construct transformation scripts and mapping files, focusing first on mapping data that is immediately useful, and reusing existing scripts as much as possible.
5. Develop test frameworks for your data mapping. This will typically include detailed tests on small portions of the transformation outputs – e.g. to ensure no unexpected URIs and no wrong property cardinalities – before completing the transformation of a whole database into RDF and testing the output in its entirety. Test frameworks are essential when this transformation process has to be repeated due to updates of the source databases.
6. Make all the scripts, configuration files, and ontologies openly accessible to the community.

The tests in Step 5 might require changes to the schema and mapping scripts. Hence, this process should be iterative. Note that the way data are structured can have significant impact on query performance and on the convenience of designing complex queries. Identifying queries to be executed in the target applications at an early stage, and including these queries as part of the tests in Step 5, will reduce the chances of being hindered by query performance.

4.2.3. Use of SPARQL endpoints

A SPARQL endpoint can be queried conveniently from scripting languages such as Perl or Python, from Web browser applications implemented in JavaScript or Flash, and soon from work flow tools like Taverna [16]. The expressive power of the SPARQL query language enables a wide range of queries to be answered, which is valuable where data query requirements are continuously emerging. We have shown that these benefits apply even without perfect ontological alignment of data from different sources, and without the need for heavyweight reasoning.

To achieve good query performance at scale, we adopted an RDF caching strategy. While software for SPARQL virtualization will improve, caching will probably still be preferable in many situations. If each data provider maintained their own SPARQL endpoint, this would spread the burden of ensuring that data are kept fresh. However, standard protocols that support change notification and cache management would also be desirable.

We hope biomedical data providers come to view provision of programmatic access to data as their top priority. The availability of expressive, performant and reliable programmatic query interfaces to those data would transform the business of biomedical data integration, and would enable others to focus on adding value by providing cross-database search, analysis and visualisation services for a range of specialized communities.

4.2.4. Data modelling

We highlight the Chado relational schema, which provides a basic framework for a range of genome-associated data types, as a valuable tool for more generic biomedical data integration. Based on the Chado relational schema, we have developed the Chado-in-OWL ontology. We have also published a mapping to RDF for FlyBase, a major implementation of the Chado relational schema, which could be adapted with little effort to other Chado databases such as GeneDB.[34]

4.2.5. A data web for medicine

In the OpenBiomed for Alternative Medicine Project,[35] the TCM-GeneDIT database,[36] a data resource about traditional Chinese medicine, has been transformed into RDF and made accessible through a SPARQL endpoint. A user-facing application of use in translational medicine has been built to link this source with four other biomedical data sources that are also accessible through SPARQL, thus integrating medical knowledge from alternative medicine and Western medical research [44]. The application is able, for example, to search for clinical trials using a given herb, or for side effects of a herb based on the chemical ingredients it contains.

This application confirms (a) the ease by which data fusion applications can be built, once data sources are published as RDF, (b) that useful integration can be achieved even without designing a full-fledged common ontology to which each dataset is aligned, although such a central ontology might be needed in the future for more complex queries, and (c) that while schema mapping is not an absolute requirement, solving the co-reference problem by mapping heterogeneous identities between data sources is of great importance. Although in the OpenBiomed for Alternative Medicine Project we did not have as complex a data identity landscape as in the OpenFlyData project, we found that interlinking data at the instance level made some of the queries simpler and perform faster.

4.2.6. A data web for classical art

To demonstrate that the same data integration principles are of generic applicability, they have also been applied in the CLAROS Project (Classical Art Research Online Services)[37] to link data from four major academic centres in Oxford, Cologne and Paris that document information about classical art objects held by the world's museums, enabling data about different types of object, for instance Greek vases held in Oxford and Greek statues held in Cologne, formerly only accessible from each source independently, to be integrated into a single user interface with sophisticated search and browse capabilities [45].

5. Conclusion

We have shown that Semantic Web standards can be combined with light-weight approaches to Web programming based on Web 2.0 patterns to enable integration of heterogeneous research data from distributed databases of gene expression data in ways that might be useful in translational bioinformatics. While initially developed for *Drosophila* bioinformatics resources, the data web technologies and patterns we have employed are generic, as we

[33] http://www.w3.org/Provider/Style/URI.

[34] http://genedb.org/.

[35] http://www.open-biomed.org.uk/for-tcm/.

[36] http://tcm.lifescience.ntu.edu.tw/.

[37] www.clarosnet.org/.

have recently demonstrated in two distinct applications, one in medicine, and the second in classical art.

Acknowledgments

The authors gratefully acknowledge the assistance of the Jena team formerly at the Hewlett Packard Research Labs in Bristol, in particular Andy Seaborne. *Funding:* This work was supported by funding of the FlyData Project by BBSRC (BB/E018068/1) and of the FlyWeb Project by the UK Joint Information Systems Committee (JISC).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jbi.2010.04.004.

References

[1] Vincent JP, Salecker I. Why flies are relevant to medical research. *Mill Hill Essays 2004*. London: MRC National Institute for Medical Research; 2004.
[2] Drosophila Board of Directors. Drosophila Board White Paper; 2007. Available from: flybase.org/static_pages/news/whitepapers/DrosBoardWP2007.pdf.
[3] Chintapalli VR, Wang J, Dow JAT. Using FlyAtlas to identify better *Drosophila melanogaster* models of human disease. Nat Genet 2007;39:715–20.
[4] Tomancak P, Beaton A, Weiszmann R, Kwan E, Shu S, Lewis SE, et al. Systematic determination of patterns of gene expression during *Drosophila* embryogenesis. Genome Biol 2002;3:81–8.
[5] Tomancak P, Berman B, Beaton A, Weiszmann R, Kwan E, Hartenstein V, et al. Global analysis of patterns of gene expression during *Drosophila* embryogenesis. Genome Biol 2007;8:R145.
[6] Zhao J, Klyne G, Miles A, Benson E, Gudmannsdottir E, White-Cooper H, et al. FlyTED: the Drosophila Testis Gene Expression Database. Nucleic Acids Res (Database Issue) 2009;38:D710–5.
[7] Barreau C, Benson E, White-Cooper H. Comet and cup genes in *Drosophila* spermatogenesis: the first demonstration of post-meiotic transcription. Biochem Soc Trans 2008;36:540–2.
[8] Arbeitman MN, Furlong EEM, Imam F, Johnson E, Null BH, Baker BS, et al. White KP Gene expression during the life cycle of *Drosophila melanogaster*. Science 2002;297:2270–5.
[9] Lécuyer E, Yoshida H, Parthasarathy N, Alm C, Babak T, Cerovina T, et al. Krause HM Global Analysis of mRNA localization reveals a prominent role in organizing cellular architecture and function. Cell 2007;131:174–87.
[10] Grumbling G, Strelets V, The FlyBase Consortium. FlyBase: anatomical data, images and queries. Nucleic Acids Res 2006;34(Database Issue):D484–8.
[11] Lyne R, Smith R, Rutherford K, Wakeling M, Varley A, Guillier F, et al. FlyMine: an integrated database for *Drosophila* and *Anopheles* genomics. Genome Biol 2007;8:R129.
[12] Goble C, Stevens R. State of the nation in data integration for bioinformatics. J Biomed Inform 2008;41:687–93.
[13] Stein LD. Towards a cyber infrastructure for the biological sciences: progress, visions and challenges. Nat Rev Genet 2008;9:678–88.
[14] Wilkinson MD, Senger M, Kawas E, Neerincx PBT, Leunissen JAM. Interoperability with Moby 1.0 – it's better than sharing your toothbrush! Brief Bioinform 2008;9:220–31.
[15] Li P, Castrillo JI, Velarde G, Wassink I, Soiland-Reyes S, Owen S, et al. Performing statistical analyzes on quantitative data in Taverna workflows: an example using R and maxdBrowse to identify differentially-expressed genes from microarray data. BMC Bioinf 2008;9:334.
[16] Hull D, Wolstencroft K, Stevens R, Goble C, Pocock MR, Li P, et al. Taverna: a tool for building and running workflows of services. Nucleic Acids Res 2006;34(Web Server issue):W729–32.
[17] Marshall MS, Prud'hommeaux E. A prototype knowledge base for the life sciences. World Wide Web Consortium (W3C) Interest Group Note; 2009. Available from: http://www.w3.org/TR/hcls-kb/.
[18] Ruttenberg A, Clark T, Bug W, Samwald M, Bodenreider O, Chen H, et al. Advancing translational research with the Semantic Web. BMC Bioinf 2007;8(Suppl. 3):S2.
[19] Feigenbaum L, Herman I, Hongsermeier T, Neumann E, Stephens S. The Semantic Web in action. Sci Am 2007;297:90–7.
[20] Gudivada RC, Qu XA, Chen J, Jegga AG, Neumann EK, Aronow BJ. Identifying disease-causal genes using Semantic Web-based representation of integrated genomic and phenomic knowledge. J Biomed Inform 2008;41:717–29.
[21] Belleau F, Nolin MA, Tourigny N, Rigault P, Morissette J. Bio2rdf: towards a mashup to build bioinformatics knowledge systems. J Biomed Inform 2008;41:706–16.
[22] Cheung KH, Yip KY, Townsend JP, Scotch M. HCLS 2.0/3.0: health care and life sciences data mashup using Web 2.0/3.0. J Biomed Inform 2008;41:694–705.
[23] Hoffmann R. A wiki for the life sciences where authorship matters. Nat Genet 2008;40:1047–51.
[24] Goble CA, De Roure DC. myExperiment: social networking for workflow-using e-scientists. In: Proceedings of the 2nd workshop on workflows in support of largescale science 25 June 2007, Monterey, California, USA. New York, NY, USA: ACM; 2007. p. 1–2. Available from: http://eprints.ecs.soton.ac.uk/15095/1/Works112v-goble2.pdf.
[25] Stevens RD, Robinson AJ, Goble CA. MyGrid: personalised bioinformatics on the information grid. Bioinformatics 2003;19:302–4.
[26] Wolstencroft K, Alper P, Hull D, Wroe C, Lord PW, Stevens RD, et al. The myGrid ontology: bioinformatics service discovery. Int. J Bioinf Res Appl 2007;3:303–25.
[27] Vandervalk BP, McCarthy EL, Wilkinson MD. Moby and Moby 2: creatures of the deep (Web). Brief Bioinform 2009;10:114–28.
[28] Zhao J, Miles A, Klyne G, Shotton D. OpenFlyData: the way to go for biological data integration. In: Proceedings of the 6th international workshop on data integration in the life sciences – DILS 2009; 20–22 July 2009. Manchester, UK: Springer; 2009. p. 47–54.
[29] Berners-Lee T. Weaving the web: the original design and ultimate destiny of the World Wide Web by its inventor. San Francisco: Harper Collins; 1999.
[30] Berners-Lee T, Hendler J, Lasilla O. The Semantic Web. Sci Am 2001;284:35–43.
[31] Shotton D. Data webs for image repositories. In: Dutton WH, Jeffreys PW, editors. World Wide research: reshaping the sciences and humanities. Cambridge: MIT Press; 2010 [Chapter 3.1]. p. 118–21, in press.
[32] Fielding RT. Architectural styles and the design of network-based software architectures. Representational state transfer (REST). Ph.D. thesis, University of California, Irvine; 2000 [Chapter 5]. Available from: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.
[33] Prud'hommeaux E, Seaborne A, editors. SPARQL query language for RDF. World Wide Web Consortium (W3C) recommendation; 2008. Available from: http://www.w3.org/TR/rdf-sparql-query/.
[34] Clark KG, Feigenbaum L, Torres E, editors. SPARQL protocol for RDF. World Wide Web Consortium (W3C) recommendation; 2008. Available from: http://www.w3.org/TR/rdf-sparql-protocol/.
[35] Merali Z, Giles J. Databases in peril. Nature 2005;435:1010–1.
[36] Campbell P. The database revolution. Funding agencies face conflicting challenges in supporting the databases essential to modern biology [Editorial]. Nature 2007;445:229–30.
[37] Klyne G, Carroll JJ, editors. Resource Description Framework (RDF): concepts and abstract syntax. World Wide Web Consortium (W3C) recommendation; 2004. Available from: http://www.w3.org/TR/rdf-concepts/.
[38] Mungall CJ, Emmert DB, the FlyBase Consortium. A Chado case study: an ontology-based modular schema for representing genome-associated biological information. Bioinformatics 2007;23:i337–46.
[39] Zhao J, Klyne G, Shotton D. Building a Semantic Web image repository for biological research images. In: Proceedingd of the 5th European Semantic Web conference, vol. 5021; 2008. p. 154–69. Available from: http://www.springerlink.com/content/t74862h1278117g6/fulltext.pdf.
[40] Bizer C, Schultz A. Benchmarking the performance of storage systems that expose SPARQL endpoints. In: Proceedings of the 4th international workshop on scalable Semantic Web Knowledge Base Systems (SSWS); 2008. Available from: http://www4.wiwiss.fu-berlin.de/bizer/pub/BizerSchulz-BerlinSPARQLBenchmark.pdf.
[41] Campbell P. Data's shameful neglect [Editorial]. Nature 2009;461:145.
[42] Samwald M, Cheung K-H, editors. Experiences with the conversion of SenseLab databases to RDF/OWL. World Wide Web Consortium (W3C) Interest Group Note; 2008. Available from: http://www.w3.org/TR/hcls-senselab/.
[43] Schimdt DC. Model-driven engineering [Editorial]. IEEE Comput 2006;39:25–31.
[44] Jentzsch A, Zhao J, Hassanzadeh O, Cheung K-H, Samwald M, Andersson B. Linking open drug data. Graz, Austria, 2009. First Prize in the iTriplification Challenge; 2009. Available from: http://blog.aksw.org/2009/triplification-challenge-2009-winners/.
[45] Kurtz D, Parker G, Shotton D, Klyne G, Schroff F, Zisserman A, et al. CLAROS – bringing classical art to a global public. In: Proceedings of the 2009 fifth IEEE international conference on e-Science (e-Science 2009), 9–11 December 2009. Oxford, UK; 2009. p. 20–7. doi:10.1109/e-Science.2009.11.